Maximizing the value of the Software Development Process by Game Theoretic Analysis

Murat Yilmaz
Lero Graduate School in Software Engineering
Dublin City University
Ireland
Murat.Yilmaz@computing.dcu.ie

Rory V. O'Connor
Dublin City University, Ireland
Lero, the Irish Software Engineering Research Centre
roconnor@computing.dcu.ie

ABSTRACT

In this paper, we introduce the novel approach of employing the economic mechanism design concept in the software development process, and investigate methods to create and adjust the incentives and disincentives of the process and align them with the motivations of the participants in order to maximize the delivered value of a software project. We consider that software development can be viewed as an economic activity inside an information exchange economy, and therefore, based on game theoretic principles, our aim is to create a people centric process model for software development.

Keywords

software development process, software engineering economics, economic mechanism design, game theory.

1. INTRODUCTION

This research is concerned with understanding the software development process as an economic mechanism, and explores ways to align the incentives and disincentives with the motivations of the participants in order to maximize the project and organizational goals, and therefore welfare of a software project.

We consider that opportunities for improving productivity in the software development effort are to be found in the human attributes associated with the activities of a software organization [8]. Software development practice is commonly conducted by teams consisting of actors identified by characteristics of individualism, rationality, and mutual interdependence [17].

Although software systems are socially interactive landscapes that are not easily specifiable in detail [34], the software development process can be characterized by spectrum of social interactions; the interaction of multiple variables and actors with different "career anchors" [4] (i.e. characteristics like skills, values, goals, motivations and preferences), making a series of investment decisions [10]. Each of these decisions are made according to goals and values of each actor which affect the expenditure of valuable resources, such as time, talent, and financial investments. An empirical study has suggested that the quality of social interactions and information exchange (e.g. knowledge sharing) among the individuals is vitally important for goal achievement [30].

Game theory is a well developed theory for describing the interactions of rational, independent agents in a variety of settings used for creating approaches in fields including, economics, computer science, social science, political science, and biology [26, 13]. Designing the rules of a game is known as *mechanism design*. In particular, an aim is to invent games that can be playable by rational players in socially beneficial ways [6]. In other words, the field of economic mechanism design is interested in creating mechanisms that improve communication and coordination in organizations to achieve a desired goal, such as allocating project resources efficiently or achieving an equitable resource distribution among the project participants [21].

In the light of these remarks, this research aims to address three main research questions: (i) how to formalize the software development process as an economic activity inside an information exchange economy; (ii) how to maximize the productivity of software development by creating incentives and disincentives; (iii) what are the limits of the approach, i.e. what is the reliability of modeling software development as an economic activity.

2. SOFTWARE PROCESS

A software process is a set or order of organizational activities (sometimes as a workflow) constrained with entrance and the exit criterions by man, machines and methods [20]. The actual goal is to provide the production of high quality products that meets the needs of its stakeholders within a balanced schedule and budget [35]. In particular, a typical software process aims to solve the potential and future problems (e.g. productivity) of software development with respect to planning and budgeting constraints.

Despite the fact that, sometimes, even it is not explicitly defined, all software organizations use a form of process related with their beliefs, values, goals, or organizational skills [27]. A broad definition for software development process should encompasses development, deployment and mainte-

nance of a software product which should include organizational structures and policies (e.g. task definitions), people in terms of their activities, technologies, and product functionalities [16].

2.1 Software Process Models

A process model prescribes the activities of development and the ways to measure and control the outcomes by understanding the steps taken throughout a process. Moreover, it concentrates on the abstract representation of the definition of states, activities, and mechanisms inside the software process by rendering descriptions of the tasks of development, their relations, and the resulting outcomes [1].

Acuna *et al.*[1] states the confusion between software process and life cycle. The life cycle usually represents how software should be developed, the product output among the phases, it should be specific and organization dependent. On the other hand, a process model presents the activities preformed for managing, developing and maintaining software systems and it should be general and project dependent.

Many different variants of development models have been created. Conventional depiction of a software process models include the waterfall model [29], the iterative enhancement model [2], and the spiral model [9]. In addition the International Organization for Standardization (ISO) has developed the ISO12207 [22] standard for software lifecycle processes, which aims to be the standard that defines all the tasks required for developing and maintaining software, but which does not imply a specific lifecycle model. Instead, processes defined this standard need to be outlined onto a development model which should be agreed within the organization [25].

Based on the authors research it is apparent that there does not exist an economic model/paradigm for the analytical modelling of the human behaviour within any software process model.

2.2 Software Process Improvement

The field of software process improvement (SPI) is established as an engineering management approach. It stems from both software engineering and field of management information systems [18]. The notion of *software process improvement* can be defined as organized set of methods or bunch of activities for improving the efficiency of software practices [1]. This improvement actually realized by application of scientific techniques to observe the improvement progress in a process, services and products [14]. One of the main goals of improvement process is to understand the underlying working mechanisms of an organization (e.g. business value creation activities) and make the organization more efficient (i.e. capable and mature) by; (i) concentrating the right activities that the organization want to do better (define a process), (ii) creating tools or methods to help the organization or individuals to do these things more efficiently (asses the process), (iii) observing the ways to improve (refine the process) in a period of time while in progress [27].

3. SOFTWARE PROCESS ENGINEERING ECONOMICS

As the field of software process engineering continues to expand, more researchers start to investigate other disciplines to determine how new methods or techniques might be applied in software engineering research. For example, economics studies how people and larger entities make choices, and the effects of these choices on individuals and environments. Similarly, in any development process, individuals and entities shape evolution of the software process by their decisions.

Game theory tackles one of the most complex analytical problems: modeling human choices, interactions, and social behaviors. The two potential uses of economic game theory in software process engineering need to be identified. Firstly, it is important to deal with the resource allocation issues which could cause project failure. These problems usually arise as social dilemmas out of the mixture of competition and cooperation among individuals. A game theoretic model may help to analyze software projects by modeling possible choices of individuals and parties which can also highlight resource allocation problems. Moreover, a model may help to investigate ways to addresses various kind of conflicting issues (e.g. managerial, technical), that surround all phases of the software process. Secondly, economic game theory offers research methods for creating and setting up organizational rules to determine ways to create incentives in order to achieve desired goals.

Although the field of software economics has been recognized as an important sub-discipline of software engineering [8, 24] for many years, software process economics primarily addresses problems of estimating the cost of programming projects, choosing a suitable pricing strategy, or appraising economic risks [24]. This approach is not adequate for two reasons: Firstly, from an economic perspective software process is considered as a value generation activity rather than a cost to be minimized. Secondly, in any software process, maximizing the decision-making effectiveness of the participants is vitally important [11]. It is now generally accepted in the software community that conflicting or poorly coordinated decisions can have acute implications on the economic viability of a software project and the resulting software product [10].

4. GAME THEORY IN SOFTWARE ENGINEERING LITERATURE

Integrating game theoretic decision support into development environments should be possible in most software development environments. Some limited approaches already exist. Lagesse [23] suggests optimizing task assignment and planning activities of projects by aligning the preferences of employees. His aim is to provide an algorithm for matching preferences of employees and tasks in software projects, and to optimize resource allocation in a software project. Another approach depicts the benefits of cooperative or competitive games by investigating social interactions [7, 17]. Cockburn [12] defines software development as a game of communication and invention with limited resources where strategic activities shape the software development process. Baskerville *et al.* [3] investigate best practices for high ve-

locity (Internet speed) software development. Their work focuses on decisions in several levels of an organization (i.e. market, portfolio and project) to explain how some of the key organizational and technical factors like time, speed, scope and resources can affect the decisions and decision makers in knowledge based organizations. Their investigation concludes that observed problems and changes in the marketplace can help to align the strategies in software development.

Conventional software decision making intends to minimize irreversible actions surrounding the development process [5]. This can be done by; (i) delaying unchangeable decisions until uncertainty is weakened [28], or (ii) investigating the environment to understand the conflicting issues. Sullivan *et al.* [31] show how to do this in a principled way by treating such decisions as investment decisions and applying the theory of options. The limit of this kind of reasoning are especially apparent with software architectural decisions which are risky and hard to reverse. Vajja and Prabhakar [33] investigate the architecture design process by modeling conflicts (that are caused by different quality attributes) as a game-theoretic problem. To the best of our knowledge no other work in the literature addresses this problem. However, there is much more to be done to optimize architecture design problems.

Several researchers have shown that social behavior problems may be identified and modelled by using Prisoner's Dilemma [26], a formal strategic game. Hazzan and Dubinsky [19] illustrate that the source of some social dilemmas in software development is due to the competitive behavior among team members, while Feijs [15] is concerned with interaction of a programmer and a software tester. It has been found that team members do not prefer to cooperate when there is no guarantee that cooperation is returned. In most of the projects they have studied, the outcome of cooperation is not known or not well defined [32]; as a matter of fact, participants usually do not trust their teammates, so they prefer to compete. This is a strong motivation for understanding software development in terms of game theory.

5. RESEARCH GOAL

The goal is to model software development process as an economic activity inside an information exchange economy to understand how social factors (e.g. motivations of the participants) influence the productivity of the development process. To this end, a game theoretic model will be created. This model will not only help us to understand the social dilemmas (e.g. interpersonal conflicts), but measures the effects of social interactions over software quality factors (e.g. social productivity of software organizations) and enable us to create strategies for efficient management of assets within software organizations.

6. PROPOSED METHODOLOGY

We will use a mixed method strategy (collection of quantitative and qualitative data and computer simulation techniques) which will be executed in three phases;

- The first step is to conduct a rigorous analysis of actual software projects and organizations to collect quantitative and qualitative data about participants characteristics to create personas (archetypes of participants). Further, we will analyze collected data to seek answers to some questions; e.g. could a person exhibit different personas in different situations? If so, does situation need to be modeled?
- The second step would be to refine the proposed economic model for software. Therefore, the data collected among the participants and situations will be classified and introduced into our economic model.
- Finally, in order to evaluate the effectiveness of our model and to observe the validity of our solution, we will perform a simulation (i.e. a social simulation game). We will virtualize a medium or a large scale software project to observe causes or effects of various situations.

7. OUTCOME OF RESEARCH

In this research, we have emphasized that the individual motivations of the participants need to be discovered and used to tune the organization and its relationship to the participants in a way that can maximize the productivity of a project as a whole.

As a technical achievement, in our research, we propose an economic model (i.e. a method for analytical modelling of human behaviour) for the software development process. This model should help us to to find ways to improve the social productivity (i.e. management of social interactions for better outputs) of software development by optimizing project resource allocation for the mutual benefit of all participants taking part in a software development organization.

Eventually, we expect to develop and test a people centric (game theoretic) software process model. The model will rely on social interactions and henceforth should include generalizable knowledge about human behaviour among software organizations which should also considered as an exploitation of a scientific achievement.

Acknowledgments

This work is supported, in part, by Science Foundation Ireland grant number 03/CE2/I303-1 to Lero, the Irish Software Engineering Research Centre (www.lero.ie).

8. REFERENCES

[1] S. T. Acuna, N. Juristo, A. M. Moreno, and A. Mon. *A Software Process Model Handbook for Incorporating People's Capabilities*. Springer-Verlag New York, Inc., 2005.

[2] V. R. Basili and A. J. Turner. Iterative enhancement: A practical technique for software development. *IEEE Transactions on Software Engineering*, 4:390–396, 1975.

[3] R. L. Baskerville, L. Levine, B. Ramesh, and J. Pries-Heje. The high speed balancing game: How software companies cope with internet speed. *Scandinavian Journal of Information Systems*, 16(1):11–54, 2004.

[4] S. Beecham, N. Baddoo, T. Hall, H. Robinson, and H. Sharp. Motivation in software engineering: A

systematic literature review. *Information and Software Technology*, 50(9-10):860–878, Aug. 2008.

[5] S. Biffl, B. Boehm, and H. Erdogmus. *Value-based software engineering*. Springer-Verlag New York Inc, 2006.

[6] K. G. Binmore. *Playing for real*. Oxford University Press US, 2007.

[7] J. Blumen. The prisoner's dilemma in software development. http://www.spectacle.org/995/sw.html.

[8] B. Boehm. *Software Engineering Economics*. Prentice Hall, Nov. 1981.

[9] B. Boehm. A spiral model of software development and enhancement. *Computer*, 21(5):61–72, 1988.

[10] B. Boehm. Value-based software engineering: reinventing. *SIGSOFT Softw. Eng. Notes*, 28(2):3, 2003.

[11] B. Boehm and K. J. Sullivan. Software economics: a roadmap. In *Proceedings of the Conference on The Future of Software Engineering*, pages 319–343, Limerick, Ireland, 2000. ACM.

[12] A. Cockburn. *Agile software development: the cooperative game*. Addison-Wesley, 2007.

[13] A. K. Dixit and S. Skeath. *Games of Strategy*. W. W. Norton & Company, June 1999.

[14] T. Dyba, T. Dingsyr, and N. B. Moe. *Process Improvement in Practice: A Handbook for It Companies (The Kluwer International Series in Software Engineering, 9)*. Kluwer Academic Publishers, 2004.

[15] L. Feijs. Prisoner dilemma in software testing. *Computer Science Reports*, 1:65–80, 2001.

[16] A. Fuggetta. Software process: a roadmap. In *ICSE '00: Proceedings of the Conference on The Future of Software Engineering*, pages 25–34. ACM, 2000.

[17] M. Grechanik and D. E. Perry. Analyzing software development as a noncooperative game. In *IEE Seminar Digests*, volume 29, 2004.

[18] B. Hansen, J. Rose, and G. Tjornehoj. Prescription, description, reflection: the shape of the software process improvement field. *International Journal of Information Management*, 24(6):457–472, Dec. 2004.

[19] O. Hazzan and Y. Dubinsky. Social perspective of software development methods: The case of the prisoner dilemma and extreme programming. In *Extreme Programming and Agile Processes in Software Engineering*, pages 74–81. Springer, 2005.

[20] W. Humphrey. *Managing the Software Process*. Addison-Wesley, 1990.

[21] L. Hurwicz and S. Reiter. *Designing economic mechanisms*. Cambridge Univ. Pr., May 2006.

[22] ISO/IEC. *Amendment to ISO/IEC 12207-2008 - Systems and software engineering â Software life cycle processes*, 2008.

[23] B. Lagesse. A Game-Theoretical model for task assignment in project management. In *2006 IEEE International Conference on Management of Innovation and Technology*, pages 678–680, Singapore, 2006.

[24] L. S. Levy. *Taming the tiger: software engineering and software economics*. Springer-Verlag New York, Inc., 1987.

[25] B. Mutafelija and H. Stromberg. *Process Improvement with CMMI v1. 2 and ISO Standards*. Auerbach Publications, 2008.

[26] M. J. Osborne and A. Rubinstein. *A course in game theory*. MIT Press, 1994.

[27] J. R. Persse. *Process Improvement Essentials*. O'Reilly Media, Inc., Sept. 2006.

[28] T. Poppendieck. *Lean software development: An agile toolkit*. Addison-Wesley Professional, 2003.

[29] W. Royce. Managing the development of large software systems. In *Proceedings of IEEE Wescon*, volume 26, 1970.

[30] S. Ryan and R. V. O'Connor. Development of a team measure for tacit knowledge in software development teams. *Journal of Systems and Software*, 82(2):229–240, Feb. 2009.

[31] K. Sullivan and P. Chalasani. Software design decisions as real options. 1997.

[32] J. E. Tomayko and O. Hazzan. *Human Aspects of Software Engineering*. Firewall Media, Dec. 2005.

[33] K. K. Vajja and P. TV. Quality attribute game: a game theory based techniquefor software architecture design. In *Proceeding of the 2nd annual conference on India software engineering conference*, pages 133–134, Pune, India, 2009. ACM.

[34] P. Wegner. Why interaction is more powerful than algorithms. *Communications of the ACM*, 40(5):80–91, 1997.

[35] S. Zahran. *Software Process Improvement: Practical Guidelines for Business Success*. Addison Wesley, 1998.